50

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/713,731	11/15/2000	Timothy D. Anderson	TI-29298	3516

23494 7590 01/12/2005

TEXAS INSTRUMENTS INCORPORATED
P O BOX 655474, M/S 3999
DALLAS, TX 75265

EXAMINER	
GERSTL, SHANE F	
ART UNIT	**PAPER NUMBER**
2183	

DATE MAILED: 01/12/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/713,731	ANDERSON, TIMOTHY D.	
	Examiner	Art Unit	
	Shane F Gerstl	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *17 December 2004 and 22 November 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 2-17 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 2-17 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 2-17 have been examined.

Papers Received

2. Receipt is acknowledged of the amendment papers where the papers have been placed of record in the file.

3. The objections to the abstract and title have not been addressed and remain as given below.

Specification

4. The abstract of the disclosure is objected to because it is too long. Correction is required. See MPEP § 608.01(b).

Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words. It is important that the abstract not exceed 150 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The form and legal phraseology often used in patent claims, such as "means" and "said," should be avoided. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title. It should avoid using phrases which can be implied, such as, "The disclosure concerns," "The disclosure defined by this invention," "The disclosure describes," etc.

5. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 2-4, 6, and 12-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kiuchi in view of Osovets (6,125,440).

8. In regard to claim 2, Kiuchi discloses

a. An instruction-programmable processor (figure 1), comprising:

i. a program memory for storing instruction opcodes (figure 1, element 101);

ii. a central processing unit (figure 1, element 100), including one or more execution units (element 105) for executing data processing instructions, and including an instruction fetch unit (figure 1, elements 103 and 106) for presenting a fetch address to the program memory for fetching therefrom an instruction opcode (column 5, line 48)corresponding to the fetch address;

iii. a loop cache (figure 1, elements 107 and 108), coupled to the instruction fetch unit, and comprising:

(1) a base address register (figure 2, repeat start address register), for storing a base fetch address;

(2) a branch cache register file (figure 1, element 108), having a plurality of storage locations (figure 3, elements 301a-c) for storing

instruction codes (column 7, line 67 to column 8, line 1) corresponding to a sequence of fetch addresses beginning with the base fetch address (stored in elements 214 of figure 2, start address registers), having a data input of the program memory, and having a data output (figures 1 and 3, element 111);

(3) a multiplexer (figure 1, element 102, selector), having a first input coupled to an output of the program memory (element 110), having a second input coupled to the data output of the branch cache register file (element 111), having a select input (element 114), and having an output coupled to the instruction fetch unit of the central processing unit (figure 1, output of multiplexer to element 103); and

(4) loop cache control logic (figure 1, elements 107 and 106 and figure 3, element 300), having a first control output coupled to a control input of the program memory (figure 1, element 122) and having a second control output coupled to the select input of the multiplexer (element 114), the loop cache control logic for controlling the multiplexer (element 102) to select the output of the branch cache register file (element 111) and for disabling a read of the program memory (column 6, lines 55-57), responsive to the fetch address corresponding to one of the instruction codes stored in the branch cache register file. [In column 6, lines 15-20, Kiuchi

shows that the address of a previously executed instruction is fetched when in a repeat operation, but the address is not generated from the program counter and thus the instruction is not fetched from memory. Therefore the instruction code stored the branch cache register file.]

iv. wherein the loop cache control logic has a third control output coupled to the branch cache register file (figure 1, element 118) for controlling writes thereto and reads therefrom (column 7, lines 40-44);

v. Kiuchi also discloses the loop cache control logic (elements 107 and 106) also that has an input for receiving a signal (element 117) indicating that a fetch address corresponds to an instruction (column 7, lines 8-9), the loop cache control logic also for controlling the branch cache register file to load an instruction code received at its data input from the program memory (as described above) responsive to receiving a signal (element 118 and column 7, lines 13-16) in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file.

b. Kiuchi discloses that the instruction which generates a signal is a repeat instruction generating a repeat signal and not a backward branch generating a backward branch signal.

c. Osovets has taught in column 10, lines 5-12, logic that uses the current and previous addresses of instructions to see if the processor branched to a

backward location. Column 9, lines 34-51 of Osovets has shown that the repeated or looped instructions are retained in the shift register (branch cache register file) once a jump back or branch backwards instructions is decoded (thus creating a backward branch signal). The included IEEE definitions of "load" shows that to load is "to place into internal storage" (definition 4), "to copy...data...to internal storage" (definition 9B), or "to enter data...into storage or working registers" (definition 13). Thus, to load is to place data into storage or registers, which is then inherently retained for some time. In order to retain a value as given in the reference, the data must be placed or loaded there.

d. Osovets has taught in column 2, line 60 – column 3, line 13 that the extra repeat instruction of use in Kiuchi can lead to programmer error when coding instructions for the processor which may further result in the loss of intended power-saving abilities and how it is desirable to be able to accommodate loops without any special programming instruction. The abilities to avoid programmer error in coding and to avoid any loss in realization of power-saving techniques would have motivated one of ordinary skill in the art to use the backwards branch and the detection thereof to indicate a loop as shown by Osovets and incorporate it into the design of Kiuchi.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kiuchi to use the method for detecting a loop taught by Osovets in order that one does not encounter programming errors and loss of power-saving realization.

9. In regard to claim 3, Kiuchi in view of Osovets discloses the processor of claim 2, as described above, further comprising:

a. backward branch detection logic (column 6, lines 11-14), [the repeat operation indicates the loop has started and that a backward branch was encountered as described in the preceding paragraph.]

b. Kiuchi in view of Osovets as previously described does not disclose that the backward branch detection logic comprises a last fetch register for storing a previous fetch address and a comparator for comparing a current fetch address to the contents of the last fetch register, the comparator having an output for generating the backward branch signal responsive to the current fetch address being less than or equal to the previous fetch address.

c. Osovets has taught backward branch detection logic comprising a last fetch register (figure 3, element 220) for storing a previous fetch address and a comparator (figure 3, element 232) for comparing a current fetch address to the contents of the last fetch register, the comparator having an output for generating the backward branch signal responsive to the current fetch address being less than or equal to the previous fetch address (column 10, lines 5-12). The last fetch register is the high address register of mention that is storing the address of an instruction previously fetched. The comparator produces a signal based on if the previous address is greater than the present address or subsequently if the present address is less than or equal to the previous. This signal then shows that operation of the processor has branched back to a previous address. This signal

is then used in conjunction with another signal offered by another comparator (figure 3, element 230) that shows if the instruction required is within range (column 10, line 11) in order to create a chip enable signal which is the backward branch signal.

d. The ability to be able to see if the required instruction is in a valid range while detecting if control has branched backwards in order to detect the validity of a loop encounter would have motivated one of ordinary skill in the art to modify Kiuchi in view of Osovets, as applied previously, in order to incorporate the backwards branch detection logic of Osovets.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kiuchi in view of Osovets, as applied above, to use the backward branch detection logic taught by Osovets so that the validity of the backward branch signal would have been realized.

10. In regard to claim 4, Kiuchi in view of Osovets discloses the processor of claim 2, as described above, further comprising:

a. indexing logic (Kiuchi, figure 2, element 217) having a second input coupled to the instruction fetch unit (element 119) to receive the fetch address therefrom, and having an output coupled to an address input of the branch cache register file (element 118a) and to the loop cache control logic (figure 3, element 300), for presenting an index value.

b. Kiuchi in view of Osovets as previously described does not disclose that the indexing logic as an index comparator having a first input coupled to the base

address register where the index value corresponds to a difference between the fetch address and the base fetch address.

c. Osovets has taught indexing logic comprised of an index comparator (figure 3, element 240), having a first input coupled to the base address register (element 220), where the index value corresponds to a difference between the fetch address and the base fetch address (column 10, lines 19-22). The high address register acts as a base address register because it holds the address of the last jump (branch) instruction (column 9, lines 47-49).

d. This addressing method presented by Osovets allows for a simple and consistent method of indexing the branch cache register. This simplicity and consistency for indexing the branch cache register would have motivate one of ordinary skill in the art to incorporate the indexing design of Osovets into Kiuchi in view of Osovets, as applied previously.

It would have been obvious to one of ordinary skill in the art to modify the design of Kiuchi in view of Osovets, as applied above, to use the indexing comparator disclosed by Osovets in order to achieve a simple and consistent method for indexing the branch cache register.

11. In regard to claim 6, Kiuchi in view of Osovets discloses the processor of claim 2, as described above, further comprising:

a. a next candidate register (Kiuchi, figure 2, element 214b), for storing a fetch address (figure 2, line 119) responsive to the loop cache control logic receiving a backward branch signal (figure 1, element 113) in combination with

the fetch address not corresponding to one of the instruction codes stored in the branch cache register file; [Column 7, lines 17-21 show that the 214b register holds an address to fetch from repeatedly, which would be in the case of a backwards branch.]

b. wherein the base address register (figure 2, element 214a) is coupled to the next candidate register (element 214b), and is for loading the contents of the next candidate register as a base fetch address responsive to the loop cache control logic receiving a backward branch signal (as described above) in combination with the fetch address corresponding to the contents of the next candidate register; [The register of element 214b holds the next branch base address for the case of nested loops and sends its data to the base address register when ready.]

c. and wherein the loop cache control logic is for controlling the branch cache register file to load an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address corresponding to the contents of the next candidate register (as described above).

12. In regard to claim 12, Kiuchi, discloses: a method of fetching instructions for execution by one or more execution units (figure 1, element 105) in a central processing unit of an instruction-programmable processor (element 100), comprising:

a. receiving a sequence of fetch addresses from an instruction fetch unit (Kiuchi, figure 1, elements 103 and 106) of the central processing unit; [In column

1, lines 8-9, Kiuchi discloses that the processor executes a sequential in order program and thus the addresses fetched are sequential.]

b. Kiuchi also discloses the above being:

i. responsive to one of the received fetch addresses corresponding to an operation (column 7, lines 8-9) and not corresponding to valid contents of a loop cache (figure 1, element 108), loading a base location of a branch cache register file (stored in elements 214 of figure 2, start address registers) with an instruction code (column 5, line 48) corresponding to the fetch address from program memory.

ii. Kiuchi does not disclose that the operation is a backward branch operation, but a repeat operation, and that once the base location is loaded a valid bit is set.

iii. Osovets has taught in column 10, lines 5-12, logic that uses the current and previous addresses of instructions to see if the processor branched to a backward location. Column 9, lines 34-51 of Osovets has shown that the repeated or looped instructions are stored in the shift register (branch cache register file) once a jump back or branch backwards instructions is decoded (thus generating a backward branch signal). Osovets has also taught in column 9, lines 61-64, that addresses stored are for valid instructions. The examiner is taking this to mean that a bit which is valid is set and that this bit corresponds to the next indexed location. Since the next instructions stored in the shift register are valid as

indicated by Osovets and as admitted by Applicant, bits which are valid are set in order to store these instructions in the registers. Subsequently, one must be able to determine the validity of an instruction from its stored address as shown in lines 63-64. This means that the valid bits must be set upon storing the address in the register.

iv. Osovets has taught in column 2, line 60 – column 3, line 13 that the extra repeat instruction of use in Kiuchi can lead to programmer error when coding instructions for the processor which may further result in the loss of intended power-saving abilities and how it is desirable to be able to accommodate loops without any special programming instruction. The valid bit allows for one to avoid processor error that would be encountered from fetching an invalid instruction address.

v. The abilities to avoid programmer error in coding, any loss in realization of power-saving techniques, and processor error from invalid instructions would have motivated one of ordinary skill in the art to use the backwards branch and the detection thereof to indicate a loop as shown by Osovets and incorporate it into the design of Kiuchi.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kiuchi to use the method for detecting a loop taught by Osovets in order that one does not encounter programming errors, loss of power-saving realization, and processor errors from invalid instruction addresses.

c. then, responsive to receiving a fetch address following the backward branch operation (Kiuchi, column 6, lines 11-24) and within a storage capacity of the branch cache register file (calculated by comparator in column 8, lines 46-55), fetching the instruction code corresponding to the fetch address from program memory (column 5, lines 46-48), loading the instruction code in a next indexed location (figure 2, element 216, creates index) of the branch cache register file (figure 2), and setting a valid bit corresponding to the next indexed location (as described above); and

d. then, responsive to receiving a fetch address corresponding to a location of the branch cache register file for which a corresponding valid bit is set:

i. disabling the program memory from performing a read access (column 6, lines 55-57);

ii. forwarding the contents of the location of the corresponding branch cache register file to the central processing unit. The output of the register file (figure 1, element 108) is selected by the selector (102) and sent through the decoder (104) to the execution unit (105).

13. In regard to claim 13, Kiuchi in view of Osovets discloses the method of claim 12, as described above, further comprising:

a. after receiving a fetch address, comparing the received fetch address with the contents of a base address register to determine a difference value (Kiuchi, column 8, lines 46-48);

b. responsive to the difference value being greater than a capacity of the

branch cache register file, determining whether the received fetch address corresponds to a backward branch relative to a previous fetch address (Ostovets, column 10, lines 5-29); [The difference value created indicates that the range is greater than the capacity of the register file (Kiuchi, column 8, lines 52-53)]

c. responsive to the determining step determining that the received fetch address corresponds to a backward branch (as described above), storing the received fetch address in the base address register (Kiuchi, column 7, lines 17-21);

d. wherein the loading step is performed responsive to the determining step determining that the received fetch address corresponds to a backward branch (as described in the previous paragraph).

14. In regard to claim 14, Kiuchi in view of Osovets discloses, as described above, the method of claim 13, wherein the determining step comprises:

a. storing a previous fetch address in a previous fetch address register (Kiuchi, figure 2, element 214a);

b. comparing the received fetch address to the contents of the previous fetch address register to determine whether the received fetch address is less than the contents of the previous fetch address register (Osovets, column 10, lines 5-29).

15. In regard to claim 15, Kiuchi in view of Osovets discloses the method of claim 12, as described above, further comprising:

a. after receiving a fetch address, detecting whether the received fetch

address corresponds to a backward branch relative to a previous fetch address (Osovets, column 10, lines 5-29); [The function referred to tells if a backward branch was detected by comparing the current and previous addresses.]

b. responsive to the detecting step detecting that the received fetch address corresponds to a backward branch, comparing the received fetch address with the contents of a candidate register; [The next candidate register given in figure 2 of Kiuchi is for the use of nested loops, therefore it stores the next loop address there for loop execution. Column 7, lines 17-21 show that the 214b register holds an address to fetch from repeatedly, which would be in the case of a backwards branch.]

c. responsive to the comparing step determining that the received fetch address does not match the contents of the candidate register, loading the received fetch address in the candidate register (Kiuchi, column 9, lines 27-32);

d. wherein the step of loading the base location of a branch cache register file is performed responsive to the comparing step detecting that the received fetch address matches the contents of the candidate register. [If the processor is in the current loop, then one must load the sequential instructions of the loop as discussed previously.]

16. Claims 5, 7-8 and 16-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kiuchi in view of Osovets as applied to claims 2-4 and 6 above, and further in view of George (4,626,988).

17. In regard to claim 5,

a. Kiuchi in view of Osovets discloses the processor of claim 4, as described above, including the branch cache register file that stores instruction codes.

b. Kiuchi in view of Osovets does not disclose a valid bit register, comprising a plurality of bit locations, each associated with one of the storage locations, each for indicating whether the contents of its associated storage location contains a valid instruction code.

c. George discloses in figure 2, a valid bit register (element 35), comprising a plurality of bit locations, each associated with one of the storage locations (element 34), each for indicating whether the contents of its associated storage location contains a valid instruction code. These valid bits show whether each entry will be valid for accessing (column 4, lines 35-37).

d. The ability to know ahead of time if an array entry is valid and usable so that instructions aren't erroneously executed (George, column 4, lines 41-42, flags are the valid bits) would have motivated one of ordinary skill in the art to incorporate the valid bit register disclosed by George into the design given by Kiuchi in view of Osovets.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kiuchi in view of Osovets to include the valid bit register and method for using it disclosed by George in order to know ahead of time whether data was valid for use or not so that instructions aren't erroneously executed.

18. In regard to claim 7, Kiuchi in view of Osovets discloses the processor of claim 6, as described above, further comprising:

a. Kiuchi in view of Osovets discloses

i. an index register (Kiuchi, figure 2, in element 216), having an output coupled to an address input of the branch cache register file. [The register is updated upon receiving each fetch address in order to index the correct storage location.]

ii. Kiuchi in view of Osovets does not disclose the index register having the output coupled to an incrementer for incrementing the contents of the index register upon the loop cache receiving each fetch address;

iii. George does disclose an index register (figure 2, element 37) having the output coupled to an incrementer (figure 2, "+1") for incrementing the contents of the index register.

iv. This method of updating the index register is quick because there is no need to derive the pointer from the program counter if the processor is simply dealing with previously stored sequential instructions. This quickness would have motivated one of ordinary skill in the art to incorporate the indexing scheme of George into Kiuchi in view of Osovets for better use of the branch cache register file.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kiuchi in view of Osovets to index the branch cache register file in the manner taught by George so that when fetching sequential instructions from the branch cache register file the index calculation is quick.

b. Kiuchi in view of Osovets as applied to claim 6, discloses

i. the branch cache register file that stores instruction codes.

ii. Kiuchi in view of Osovets does not disclose a valid bit register, comprising a plurality of bit locations, each associated with one of the storage locations, each for indicating whether the contents of its associated storage location contains a valid instruction code.

iii. George discloses in figure 2, a valid bit register (element 35), comprising a plurality of bit locations, each associated with one of the storage locations (element 34), each for indicating whether the contents of its associated storage location contains a valid instruction code. These valid bits show whether each entry will be valid for accessing (column 4, lines 35-37).

iv. The ability to know ahead of time if an array entry is valid and usable so that instructions aren't erroneously executed (George, column 4, lines 41-42, flags are the valid bits) would have motivated one of ordinary skill in the art to incorporate the valid bit register disclosed by George into the design given by Kiuchi in view of Osovets.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify Kiuchi in view of Osovets to include the valid bit register and method for using it disclosed by George in order to know ahead of time whether data was valid for use or not so that instructions aren't erroneously executed.

c. wherein the loop cache control logic also has an input for receiving a sequential fetch signal indicating that the fetch address from the instruction fetch unit is in sequence with a previous fetch address; [In column 6, lines 11-24, Kiuchi has shown that the control signal 113 is sent to the program counter to show that the repeat operation has started. This means that the base address and the repeat module (set of instructions) must be stored as explained. This module is a set of sequential instructions.]

d. and wherein the loop cache control logic is for controlling the branch cache register file to store an instruction code received at its data input from the program memory (as described previously as being disclosed by Kiushi), at a storage location indicated by the contents of the index register (George, column 4, lines 24-26), responsive to receiving the sequential fetch signal (this signal indicates to store the sequence of instruction as described above) and to the bit location of the valid bit register (George, figure 2, element 35) indicating that the storage location corresponding to the contents of the index register does not contain a valid instruction code. [If the valid bit register previously disclosed does not have valid data, then that position in storage is free.]

19. In regard to claim 8, Kiuchi in view of Osovets in further view of George discloses the processor of claim 7, as described above, wherein the loop cache control logic is also for controlling the branch cache register file to present, at its output, the contents of a storage location (Kiuchi, figure 3, element 111) indicated by the contents of the index register (element 118a), responsive to receiving the sequential fetch signal and to the

bit location of the valid bit register indicating that the storage location corresponding to the contents of the index register contains a valid instruction code. [If the valid bit is not valid then the instruction will not be outputted so as to avoid error as described previously. The sequential signal described previously to show when instructions were sequential for the purpose of storing instructions to the register file serves just as vital a role for the purpose of reading from the register file so as to know when the end of the loop has been reached.]

20. In regard to claim 16, Kiuchi in view of Osovets discloses the method of claim 15, as described above, further comprising:

a. responsive to the comparing step determining that the received fetch address matches the contents of the candidate register (as described above), setting an index register (Kiuchi, figure 2, element 216); then, responsive to receiving a next fetch address, repeating the detecting step (so as to know hat state to go to next);

c. responsive to the detecting step detecting that the received fetch address does not correspond to a backward branch, determining whether the received fetch address corresponds to a sequential fetch; [In column 6, lines 11-24, Kiuchi has shown that the control signal 113 is sent to the program counter to show that the repeat operation has started. This means that the base address and the repeat module (set of instructions) must be stored as explained. This module is a set of sequential instructions. Therefore, by determining this control signal, whether a sequential fetch is encountered is determined.]

d. and responsive to the determining step determining that the received fetch address corresponds to a sequential fetch, testing a valid bit corresponding to the contents of the index register; [The valid bit is used as described above and inherently corresponding to the index register which references the instruction code.]

e. wherein the step of fetching the instruction code is performed responsive to the resulting of the testing step indicating that the valid bit corresponding to the contents of the index register is not set; [As described above, the valid bit is set upon storing. Therefore if the bit is not set, the storage location is free for an address to be fetched.]

f. wherein the disabling and forwarding steps are performed responsive to the result of the testing step indicating that the valid bit corresponding to the contents of the index register is set; [As described previously, the disabling and forwarding means that instructions will be fetched from the loop cache. Also as described before, the valid bit shows if there are valid instructions stored in response to a backward branch. As shown above, when this happens the loop cache is selected for fetching. When selected, the memory is also disabled and the instructions will be forwarded to the execution unit as described above.]

g. Kiuchi in view of Osovets, as applied to claim 15, discloses

i. an index register (Kiuchi, figure 2, in element 216) [The register is updated upon receiving each fetch address in order to index the correct storage location.]

ii. Kiuchi in view of Osovets does not disclose the index register being incremented after fetching or forwarding;

iii. George does disclose an index register (figure 2, element 37) having the output coupled to an incrementer (figure 2, "+1") for incrementing the contents of the index register.

iv. This method of updating the index register is quick because there is no need to derive the pointer from the program counter if the processor is simply dealing with previously stored sequential instructions. This quickness would have motivated one of ordinary skill in the art to incorporate the indexing scheme of George into Kiuchi in view of Osovets for better use of the branch cache register file.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kiuchi in view of Osovets to index the branch cache register file in the manner taught by George so that when fetching sequential instructions from the branch cache register file the index calculation is quick.

21. In regard to claim 17, Kiuchi in view of Osovets in further view of George discloses the method of claim 16, as described above, further comprising: responsive to the determining step determining that the received fetch address does not correspond to a sequential fetch (as described above), fetching the instruction code corresponding to the fetch address from program memory. [It has been shown that if the instruction is sequential, then the fetching takes place from the loop cache. This means that if it is

not sequential, the fetching is from another source. The only other source is the program memory.]

22. Claims 9-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kiuchi in view of Hennessy.

23. In regard to claim 9, Kiuchi discloses the processor of claim 2, wherein the program memory comprises:

a. A level one instruction memory (column 2, lines 34-35) having a read control input (column 6, lines 50-53) coupled to the loop cache control logic (figure 1, element 122), and having a data output coupled to the multiplexer (figure 1, element 110).

b. Kiuchi does not disclose a level one tag memory, for storing tag addresses corresponding to memory locations for which the level one instruction memory stores instruction codes; and a level one tag comparator, having an input for receiving the fetch address from the fetch instruction unit, and having an input coupled to the level one tag memory, for comparing the fetch address to the tag addresses to determine whether the fetch address corresponds to a memory location for which the level one instruction memory stores a valid instruction code.

c. Hennessy has taught a level one tag memory, for storing tag addresses corresponding to memory locations for which the level one instruction memory stores instruction codes (page 549, figure 7.7); and a level one tag comparator (figure 7.7), having an input for receiving the fetch address from the fetch

instruction unit, and having an input coupled to the level one tag memory, for comparing the fetch address to the tag addresses to determine whether the fetch address corresponds to a memory location for which the level one instruction memory stores a valid instruction code. Hennessy shows that these parts together comprise a cache. The address shown in the figure corresponds to an address from the fetch unit.

d. Hennessy teaches on page 542, figure 7.1 that faster memory is located closer to the processor. On page 545, Hennessy discloses that a cache is between the processor and main memory; hence a cache is faster than main memory.

e. This decreased memory access time would have motivated one of ordinary skill in the art to incorporate the level one cache system disclosed by Hennessy into the design of Kiuchi.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kiuchi to incorporate the cache system taught by Hennessy so that access times to memory are lower.

24. In regard to claim 10,

a. Kiuchi in view of Hennessy, as applied above, discloses the processor of claim 1, as described above

b. Kiuchi in view of Hennessy, as applied to claim 9, does not disclose that the program memory further comprises a level two cache, coupled to the level one instruction memory for storing instruction codes.

c. Hennessy has taught a level two cache, coupled to the level one instruction memory for storing instruction codes. On page 579, paragraph 2, Hennessy has taught that by allowing a larger secondary (level two) cache to handle misses to the primary cache one can reduce miss penalty.

d. This reduction of miss penalty would have motivated one of ordinary skill in the art to modify the design of Kiuchi in view of Hennessy, as applied to claim 9, to incorporate a level two cache as taught by Hennessy.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kiuchi in view of Hennessy, as applied to claim 9, to use a level two cache as taught by Hennessy in order to reduce the miss penalty of the cache.

25. In regard to claim 11, Kiuchi in view of Hennessy discloses the processor of claim 10, as described above, wherein the program memory and the level two cache are located on the same integrated circuit as the central processing unit. Kiuchi shows in figure 1, the data processing unit, 100, that includes the discussed embodiments all on the same unit, and thus the same integrated circuit.

Response to Arguments

26. Applicant's arguments filed 22 November 2004 have been fully considered but they are not persuasive.

27. Applicant argues that the prior art of record does not teach that the loop cache control logic is also for controlling the branch cache register file to *load* an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address not corresponding to one

of the instruction codes stored in the branch cache register file. It is specifically argued that while "to store" in the previous amendment could be interpreted as "retaining", which is admitted to being taught by the reference, the action verb "to load" cannot be met by the disclosed action of retaining. After careful reconsideration and search it has been found that "to load" can also be interpreted as "retaining" as disclosed in the prior art. The included IEEE definitions of "load" shows that to load is "to place into internal storage" (definition 4), "to copy...data...to internal storage" (definition 9B), or "to enter data...into storage or working registers" (definition 13). Thus, to load is to place data into storage or registers, which is then inherently retained for some time. In order to retain a value as given in the reference, the data must be placed or loaded there. Further, Applicant has stated at the end of the last paragraph of page 12 that instruction codes are loaded by Osovets.

28. Applicant argues that Osovets teaches the loading of every instruction code into its shift register, regardless of whether a backward branch has been taken, and the retaining of those instruction codes upon detecting such a backward branch and thus the loading is not responsive to a backward branch. The Examiner finds that each instruction code is loaded unconditionally but asserts that the instructions codes are loaded in response to encountering them. Thus upon encountering any instruction its code is loaded. Upon encountering a backwards branch instruction then, the instruction code is loaded. The claim language does not limit the scope to say that only backwards branch instructions are loaded and all other instructions are not loaded.

29. Applicant then argues that the proposed advantages of claim 2 negate suggestion from the prior art to modify its teachings in such a manner as to reach the claim. The Examiner respectfully submits that the advantages of Applicants claim have no bearing on the motivations used to combine prior art. The fact that applicant has recognized another advantage which would flow naturally from following the suggestion of the prior art cannot be the basis for patentability when the differences would otherwise be obvious. See *Ex parte Obiaya*, 227 USPQ 58, 60 (Bd. Pat. App. & Inter. 1985).

Conclusion

30. The following is text cited from 37 CFR 1.111(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Shane F Gerstl whose telephone number is (571) 272-4166. The examiner can normally be reached on M-F 6:45-4:15 (First Friday Off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Shane F Gerstl
Examiner
Art Unit 2183

SFG
January 10, 2005

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100